

FIG. 1

09541774.040300

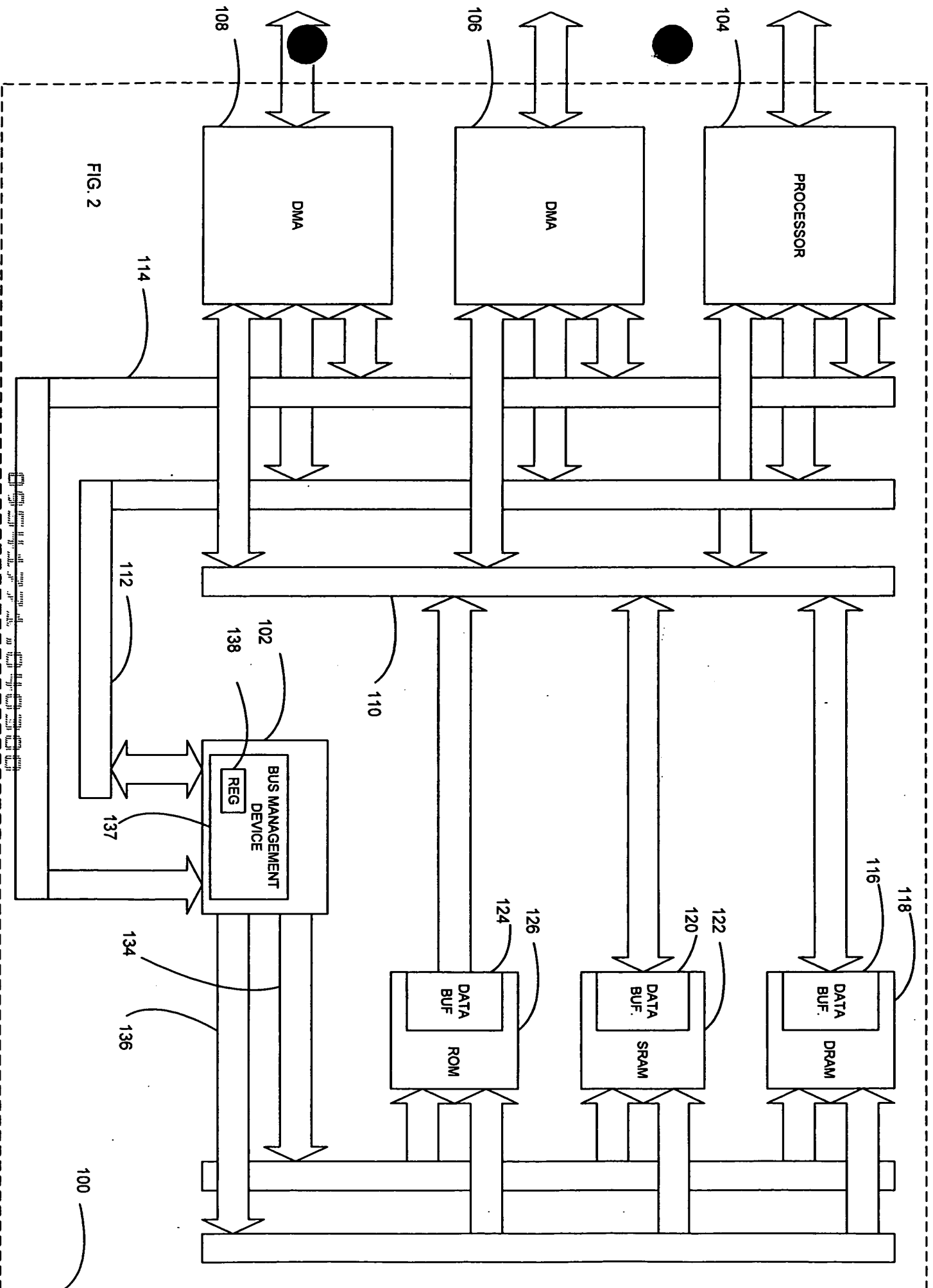




FIG. 4


```

// *****
// Convert Trob into bit stream.
// *****

always @(Trob0 or Len0)
begin
  case (Trob0) // synopsys full_case parallel_case
    2'h0: Busy0[3:0] = 4'b0001;
    2'h1: case (Len0[1:0]) // synopsys full_case parallel_case
      2'h0: Busy0[3:0] = 4'b0001;
      2'h1: Busy0[3:0] = 4'b0011;
      2'h2: Busy0[3:0] = 4'b0011;
      2'h3: Busy0[3:0] = 4'b0011;
    endcase
    2'h2: case (Len0[1:0]) // synopsys full_case parallel_case
      2'h0: Busy0[3:0] = 4'b0001;
      2'h1: Busy0[3:0] = 4'b0011;
      2'h2: Busy0[3:0] = 4'b0111;
      2'h3: Busy0[3:0] = 4'b0111;
    endcase
    2'h3: case (Len0[1:0]) // synopsys full_case parallel_case
      2'h0: Busy0[3:0] = 4'b0001;
      2'h1: Busy0[3:0] = 4'b0011;
      2'h2: Busy0[3:0] = 4'b0111;
      2'h3: Busy0[3:0] = 4'b1111;
    endcase
  endcase
end

```

FIG. 5B

```

// *****
// Convert Trob into bit stream.
// *****

always @(Trob1 or Len1)
begin
  case (Trob1) // synopsys full_case parallel_case
    2'h0: Busyl[3:0] = 4'b0001;
    2'h1: case (Len1[1:0]) // synopsys full_case parallel_case
      2'h0: Busyl[3:0] = 4'b0001;
      2'h1: Busyl[3:0] = 4'b0011;
      2'h2: Busyl[3:0] = 4'b0011;
      2'h3: Busyl[3:0] = 4'b0011;
    endcase
    2'h2: case (Len1[1:0]) // synopsys full_case parallel_case
      2'h0: Busyl[3:0] = 4'b0001;
      2'h1: Busyl[3:0] = 4'b0011;
      2'h2: Busyl[3:0] = 4'b0111;
      2'h3: Busyl[3:0] = 4'b0111;
    endcase
    2'h3: case (Len1[1:0]) // synopsys full_case parallel_case
      2'h0: Busyl[3:0] = 4'b0001;
      2'h1: Busyl[3:0] = 4'b0011;
      2'h2: Busyl[3:0] = 4'b0111;
      2'h3: Busyl[3:0] = 4'b1111;
    endcase
  endcase
end

// *****
// Shift right by 1. Shift in zero for MSB.
// FIX: the high order bit should be shifted in as zero in the Next_tdatardy block.
// *****

always @(posedge Clk or negedge nRst)
begin
  if (!nRst)
  begin
    Tdatardy <= 15'h0000;
    Mask <= 15'h0000;
  end
  else if (RWCmd0 & !RWCmd1)
  begin
    Tdatardy[13:0] <= Next_tdatardy0;
    Tdatardy[14] <= 0;

    Mask[13:0] <= Next_mask0;
    Mask[14] <= 0;
  end
  else if (RWCmd1 & !RWCmd0)
  begin
    Tdatardy[13:0] <= Next_tdatardy1;
    Tdatardy[14] <= 0;

    Mask[13:0] <= Next_mask1;
    Mask[14] <= 0;
  end
  else // shift
  begin
    Tdatardy[13:0] <= Tdatardy[14:1];
    Tdatardy[14] <= 0;
  end
end

```

FIG. 5C

COPYRIGHT 1999 HEWLETT PACKARD COMPANY

005040"040300


```

// *****
// Determine if new access can issue R/W command.
// *****

always @(Tdatardy or Mask or Csd1 or Sync1 or Tread1)
begin
  if (!Sync1)
  begin
    Cmd_rdy1      = ~(|(Mask[14:0]));
    Next_rdy1     = ~(|(Mask[14:0]));
  end
  else if (Tread1)
  begin
    // Read
    // synopsys parallel_case
    case (Csd1)
    3'h1: begin
      Cmd_rdy1 = ~(|(Tdatardy[14:5]));
      Next_rdy1 = ~(|(Mask[14:5]));
    end
    3'h2: begin
      Cmd_rdy1 = ~(|(Tdatardy[14:6]));
      Next_rdy1 = ~(|(Mask[14:6]));
    end
    3'h3: begin
      Cmd_rdy1 = ~(|(Tdatardy[14:7]));
      Next_rdy1 = ~(|(Mask[14:7]));
    end
    3'h4: begin
      Cmd_rdy1 = ~(|(Tdatardy[14:8]));
      Next_rdy1 = ~(|(Mask[14:8]));
    end
    3'h5: begin
      Cmd_rdy1 = ~(|(Tdatardy[14:9]));
      Next_rdy1 = ~(|(Mask[14:9]));
    end
    3'h6: begin
      Cmd_rdy1 = ~(|(Tdatardy[14:10]));
      Next_rdy1 = ~(|(Mask[14:10]));
    end
    default: begin
      Cmd_rdy1 = ~(|(Tdatardy[14:5]));
      Next_rdy1 = ~(|(Mask[14:5]));
    end
  endcase
end
else
  // Write
  begin
    Cmd_rdy1 = ~(|(Tdatardy[14:2]));
    Next_rdy1 = ~(|(Mask[14:2]));
  end
end
end

```

FIG. 5E

```

// *****
// Determine if new access can issue R/W command.
// *****

always @(Tdatardy or Mask or Csd0 or Sync0 or Tread0)
begin
    if (!Sync0)
    begin
        Cmd_rdy0      = ~(|(Mask[14:0]));
        Next_rdy0     = ~(|(Mask[14:0]));
    end
    else if (Tread0)          // Read
    begin
        case (Csd0)          // synopsys parallel_case
        3'h1: begin
            Cmd_rdy0 = ~(|(Tdatardy[14:5]));
            Next_rdy0 = ~(|(Mask[14:5]));
        end
        3'h2: begin
            Cmd_rdy0 = ~(|(Tdatardy[14:6]));
            Next_rdy0 = ~(|(Mask[14:6]));
        end
        3'h3: begin
            Cmd_rdy0 = ~(|(Tdatardy[14:7]));
            Next_rdy0 = ~(|(Mask[14:7]));
        end
        3'h4: begin
            Cmd_rdy0 = ~(|(Tdatardy[14:8]));
            Next_rdy0 = ~(|(Mask[14:8]));
        end
        3'h5: begin
            Cmd_rdy0 = ~(|(Tdatardy[14:9]));
            Next_rdy0 = ~(|(Mask[14:9]));
        end
        3'h6: begin
            Cmd_rdy0 = ~(|(Tdatardy[14:10]));
            Next_rdy0 = ~(|(Mask[14:10]));
        end
        default: begin
            Cmd_rdy0 = ~(|(Tdatardy[14:5]));
            Next_rdy0 = ~(|(Mask[14:5]));
        end
    endcase
    end
    else
        // Write
    begin
        Cmd_rdy0 = ~(|(Tdatardy[14:2]));
        Next_rdy0 = ~(|(Mask[14:2]));
    end
end

```

FIG. 5G

```

// *****
// Generate next bit stream by concatenating old bit stream with new bit stream.
// Bit stream includes a right shift of one bit.
// *****

always @(Tdatardy or Csd1 or Busyl or Sync1 or Tread1 or Mask1)
begin
  if (Sync1 && Tread1)                                // Read
  begin
    casex (Csd1)                                       // synopsys parallel_case
    3'h1: begin
      Next_tdatardy1 = {6'h0, Busyl, Tdatardy[4:1]};
      Next_mask1     = {5'h0, Mask1, Tdatardy[4:1]};
    end
    3'h2: begin
      Next_tdatardy1 = {5'h0, Busyl, Tdatardy[5:1]};
      Next_mask1     = {4'h0, Mask1, Tdatardy[5:1]};
    end
    3'h3: begin
      Next_tdatardy1 = {4'h0, Busyl, Tdatardy[6:1]};
      Next_mask1     = {3'h0, Mask1, Tdatardy[6:1]};
    end
    3'h4: begin
      Next_tdatardy1 = {3'h0, Busyl, Tdatardy[7:1]};
      Next_mask1     = {2'h0, Mask1, Tdatardy[7:1]};
    end
    3'h5: begin
      Next_tdatardy1 = {2'h0, Busyl, Tdatardy[8:1]};
      Next_mask1     = {1'h0, Mask1, Tdatardy[8:1]};
    end
    3'h6: begin
      Next_tdatardy1 = {1'h0, Busyl, Tdatardy[9:1]};
      Next_mask1     = {Mask1, Tdatardy[9:1]};
    end
    default: begin
      Next_tdatardy1 = 14'bxx_xxxx_xxxx_xxxx;
      Next_mask1     = 14'bxx_xxxx_xxxx_xxxx;
    end
  endcase
end
else if (Sync1 && !Tread1)                             // Write
begin
  Next_tdatardy1 = {9'h00, Busyl, Tdatardy[1]};
  Next_mask1     = {8'h00, Mask1, Tdatardy[1]};
end
else                                                     // Async
begin
  Next_tdatardy1 = 14'h0000;
  Next_mask1     = 14'h0000;
end
end

```

FIG. 5F